the agile
monkeys.

API Test
-
The CRM Service

API Test - The CRM service

Welcome to the second phase of The Agile Monkeys recruitment process!

In this phase we'll ask you to build a synthetic API project using your technology of choice. You can use any programming language or tool, but we recommend you to use the one you feel more comfortable with.

The objective is to create a REST API to manage customer data for a small shop. It will work as the backend side for a CRM interface that is being developed by a different team. As the lead developer of the backend project, you'll be in charge of the API design and implementation. Here are the requirements for the API:

- The API should be only accessible by a registered user by providing an authentication mechanism.
- A user can only:
  - List all customers in the database.
  - Get full customer information, including a photo URL.
  - Create a new customer:
    - A customer should have at least name, surname, id and a photo field.
    - Name, surname and id are required fields.
    - Image uploads should be able to be managed.
    - The customer should have a reference to the user who created it.
  - Update an existing customer.
    - The customer should hold a reference to the last user who modified it.
  - Delete an existing customer.
- An admin can also:
  - Manage users:
    - Create users.
    - Delete users.
    - Update users.
    - List users.
    - Change admin status.

The specific design is up to you, but we recommend you to take advantage of tools and/or libraries provided by your platform of choice.

You will be invited to a Slack channel where you will be in touch with our team, we encourage candidates to explain why you chose a specific framework, library or approach! Please ask them any questions you may have.

Also, please create a git repository for the project in Github or Bitbucket and share the link with the team in the Slack channel.

What we are evaluating in this test:

- Required:
    - Good code quality: Readability and simplicity, good semantics, idiomatic code and adoption of framework standards. **Write your code as if you were to publish it to production.**
    - Good software architecture: Low coupling, ease to change, good use of design patterns, use of framework or specific language patterns.
    - Good communication with the team.
    - Basic security measures (Authentication, Authorisation, SQL injection and XSS prevention).
    - Completeness and correctness of the solution.
- Extra points:
    - Good README file with a getting started guide.
    - Tests implemented for the solution.
    - Making project set-up easier for newcomers.
    - The application follows the twelve-factor app principles (https://12factor.net) in order for it to be scalable.
    - Follow OAuth 2 protocol for authentication (You can use a third party public OAuth provider).
    - The project is ready for Continuous Deployment using a provider (e.g. AWS).
    - The project uses Docker, Vagrant or other tools to make it easier to configure development environments.

Good luck!

The Agile Monkeys.

the agile
monkeys.

contact us.

theagilemonkeys.com